Software Engineering meets Scientific Computing
- Group Projects in CSE Education -

Ioan Lucian Muntean
TUM
Department of Informatics
Chair of Scientific Computing

Workshop SCSE07
Uppsala, August 15, 2007
Outline

• Motivation

• Software Team Projects

• An Example from Fluid-Structure Interaction: PeGSI

• Experiences and lessons learned
Motivation

- **The bottlenecks** in scientific computing: **hardware** and **algorithms**
  - software as change *in philosophy*
- Cooperation of the community with computer scientists in different areas of software engineering (SE)
  - verification: “do we solve the equation correctly?”
  - validation: “do we solve the right equation?”
  - formal methods, model checking
  - **software engineering mainstreaming** (create a background on SE: software development process, from specification to test and delivery)
  - **teamwork in software projects** (a step away from the classical CSE approach 1 researcher – 1 task/small project)
  - SE “best practices”
- The majority in the CSE community – **still reluctant to adopt SE practices** in the development of simulation software
- We consider extremely important
  - application performance (CPU, memory), hardware awareness, TFLOPS
  - but also maintainability, extensibility, flexibility
- A new software crisis
Motivation (cont’d)

• Alarm signals triggered
  – by the **user community**
    “The field has reached a threshold at which better organization becomes crucial. New methods of verifying and validating complex codes are mandatory if computational science is to fulfil its promise for science and society.” (Doug Post and Lawrence Votta)
    from *Physics Today*, vol. 58, no. 1, 2005
  – by the **CSE community**
    “Today’s computational science ecosystem is unbalanced, with a software base that is inadequate to keep pace with and support evolving hardware and application needs.”
    from the executive summary of the *PITAC report*, 2005
  – different reactions at different levels
    “How can I program a configuration manager as yours?”

• Our efforts for the cooperation CSE – SE materialized in two directions
  – CSE masters programme since 2000: the first one and for long time the only one – SE lecture given by a SE professor and with content trimmed to CSE curriculum
  – practical experience with software development in team projects – *learning by doing*
Software Team Projects

• Established in the “Software Engineering” diploma programme in Stuttgart (i.e. without any CSE context)
• Adopted for and adapted to the CSE/BGCE programme in 2005
  – CSE@TUM has a mandatory Software Engineering course from the very beginning
  – BGCE: Bavarian Graduate School of Computational Engineering, honours programme in the Elite Network of Bavaria, joint activity of CSE@TUM, come.tum, and CE@FAU)
  – the software project is one third of the BGCE curriculum
• Main features (TUM version)
  – during second and third semester of the master’s studies
  – team of 4-8 students
  – 6-9 months, 10 ECTS credits
  – goal: produce a complete piece of software
    • from the idea to the product
  – “pseudo-industrial” setting: customer, roles (project manager, experts, …), fines for breach of contract, …
Software Team Projects (cont’d)

- CSE-related topics so far
  - Computational Steering (2004)
  - Molecular Dynamics (2005)
  - Visualization; Computational Finance (2007)
- Topics related to the interest of the potential participants both students and researchers
- The project itself – a decision factor for joining the BGCE programme
PeGSI – Team Project on Fluid-Structure Interaction

- Software team project in 2006/2007
- Topic related to and integrated with current research at the Chair (CFD, FSI, and simulation scenario)
- **Software Requirements Specification** document

**Topic and Tasks:**
- PeGSI – Peano Geometry Sophisticated Interface
- efficient handling of geometries as needed in FSI
  - on-the-fly features, important for partitioned approach
  - software to be used in coupling environment FSI☆ce
- interface to two existing flow solvers
- development of a simple structural solver

**Requirements:**
- provide support for complex and changing geometries
- geometry interfaces to different sources (OpenCASCADE, e.g.)
- address the issue of diverging needs for the codes to be connected
- software quality, non-functional requirements, schedule and deadlines
More about PeGSI

**Implementation:**
- May 2006 – March 2007, TUM
- 4 participants (CSE/BGCE master)
- 4 countries: China, El Salvador, UA Emirates, Germany
- different background: Informatics, IndEng, CivEng
- internal customer (TUM)
- advisors

**Roadmap:**
- kick-off meeting (students, customer, advisory)
- acceptance of the specification and project’s scope
- three releases, review meetings with the customer
- contract on scope of extensions
- acceptance test
- feedback round
- detailed experience report (customer, scientific advisor)

**Customer:**
- acted as a software company – library used for the development of CFD/FSI simulation codes
- special touch to the project (documentation, modules, API, deadlines)
PeGSI Results

**CSE-related:**
- use of external software
  - for geometry handling primitives (Open CASCADE) and visualization (Paraview)
  - interface to Peano flow solver
- library for geometry handling (2D) needed in FSI simulations: support for dynamic adaptivity of the mesh, for moving/changing objects (partially)
- simple structural solver

**Software Engineering:**
- deliverables, modular code
- *all in the code* documentation
- unit tests (partially), acceptance test (Drift Ratchet geometry)
- *not all customer requirements have been fulfilled*

**Soft skills:**
- experience teamwork over a period of 9 months, know-how sharing
- project planning, time and resource management, communication, presentation
- tensions everywhere – within the team members (lack of productivity), with the advisors (workload), with the customer (reduction of requirements), mediated meetings

**Outreach:**
- presentations – Research Day
- poster – ASIM Workshop 2007, Aachen
Experiences and Lessons Learned

- The company – customer model is the driving force of this kind of projects (autonomy, contract, responsibilities, fines, rewards).
- Precise software specs at the beginning of project foster understanding project’s scope, and a realistic estimation of workload.
- The freedom to choose the implementation strategy and to select appropriate algorithms is motivating but also critical for the scientific success of the project.
- The project idea has a very positive impact on CSE/BGCE students (volunteers - v)
  - now two running projects: visualization (4+2v), computational finance (1+2v)

- To increase the team acceptance for specific SE topics (documentation, testing), a closer coordination is necessary (mini-workshops).
- A customer too deeply involved could (counterproductively) influence the development process (beyond the scope of the specification).
- Ensuring the know-how and experience transfer from one project to the next one is vital for both students and advisors (“success story”, “learning from disaster”).
Closing Remarks

• **Software team projects** – a concept valuable for CSE education
• A frame for *implementing/experimenting SE practices within the CSE context*
• *Learning by doing*, backed-up by input and feedback
• *Experience sharing* within the team and with the coming generation
• *Practical soft skills*, such as communication, presentation, or time management
• Topics for such projects – *in every CSE branch*
• *Balance* between the CSE and SE content of the project

*We know it’s not perfect yet, but we’re working on it and are open to suggestions!*

**Thanks for your attention!**